Efficient Evaluation of Shortest Travel-time Path Queries in Road Networks by Optimizing Waypoints in Route Requests through Spatial Mashups

Detian Zhang[1], Chi-Yin Chow[2], Qing Li[2], and An Liu[3]

[1] School of Digital Media, Jiangnan University, Wuxi, China
[2] Department of Computer Science, City University of Hong Kong, Hong Kong
[3] School of Computer Science, Soochow University, Suzhou, China
detian.cs@gmail.com, {chiychow,itqli}@cityu.edu.hk, anliu@suda.edu.cn

Abstract. In the real world, the route with the shortest travel time in a road network is more meaningful than that with the shortest network distance for location-based services (LBS). However, not every LBS provider has adequate resources to compute/estimate travel time for routes by themselves. A cost-effective way for LBS providers to estimate travel time for routes is to issue external requests to Web mapping services (e.g., Google Maps, Bing Maps, and MapQuest Maps). Due to the high cost of processing such external requests and the usage limits of Web mapping services, we take the advantage of direction sharing and waypoints supported by Web mapping services to reduce the number of external requests and the query response time for shortest travel-time route queries in this paper. We model the problem of selecting the optimal waypoints for an external route request as finding the longest simple path in a weighted bipartite digraph. As it is a NP-complete problem, we propose a greedy algorithm to find the best set of waypoints in an external route request. We evaluate the performance of our approach using real Web mapping services, a real road network, real and synthetic data sets. Experimental results show the efficiency, scalability, and applicability of our approach.

Keywords: Spatial mashups, Location-based services, Direction sharing, Waypoints, Web mapping services

1 Introduction

A spatial (or GIS/mapping) mashup [1–4] provides a cost-effective way for a Web or mobile application that combines data, representation, and/or functionality from at least one Web mapping service and other local/external services to create a new application. It becomes more and more popular along with the development of Web mapping services and the ubiquity of Internet access and GPS-enabled mobile devices. Typical Web mapping services are Google Maps, MapQuest Maps, Microsoft Bing Maps, Yahoo! Maps, and Baidu Maps. Based on the latest statistics of Programmable Web [5], the spatial mashup is the

most popular one among all types of mashups including search mashups, social mashups, etc.

In the real world, the route with the shortest travel time (e.g., driving, walking and cycling time) in a road network is more meaningful than the route with the shortest network distance for location-based services (LBS). Since travel time is highly dynamic due to many realistic factors, e.g., heterogeneous traffic conditions and traffic accidents, it is difficult for an LBS provider to perform travel route estimation/computation because of huge deployment cost. However, it is not a big problem for Web mapping service providers, as they have adequate resources to collect data from historical traffic statistics and/or continuously monitor the real-time traffic in road networks; besides, most of existing Web mapping services provide user-friendly APIs for applications to access travel route information, e.g., the Google Maps Directions API [6] and the MapQuest Directions Web Service [7]. Therefore, a typical and popular application scenario using spatial mashups is that an LBS provider subscribes the travel route information based on live traffic conditions from Web mapping services through their APIs to answer location-based queries for its own users [1–4].

However, retrieving the travel route information through spatial mashups suffers from the following two critical limitations [1, 2]: (1) It is costly to access travel route information from a Web mapping service, e.g., retrieving travel time from the Microsoft MapPoint Web service to a database engine takes 502 ms while the time needed to read a cold and hot 8 KB buffer page from disk is 27 ms and 0.0047 ms, respectively [8]. (2) There is a charge on the number of requests issued to a Web mapping service, e.g., the Google Maps Directions API [6] allows only 2,500 requests per day for evaluation users and 100,000 requests per day for business license users [9]. An LBS provider needs to pay to have higher usage limits. Therefore, when an LBS provider endures high workload, e.g., a large number of concurrent user queries, it needs to issue a large number of external Web mapping requests, which not only result in high business operation cost, but also induce long response time to querying users.

In this paper, we aim to explore a direction (or route/path) sharing optimization and select an appropriate set of waypoints in *external route requests* sent to a Web mapping service to reduce the number of external request and the response time to querying users for LBS providers. Given user u_i's *shortest travel-time path query* $R(o_i \rightarrow d_i)$, the basic idea of the direction sharing optimization is that the route information from query origin o_i to query destination d_i can be shared with another user u_j's query $q_j = (o_j, d_j)$ if both of its origin o_j and destination d_j are located in $R(o_i \rightarrow d_i)$. To reflect the opportunity of sharing the direction information of an external route request, we also formally define the *sharing ability* of a shortest travel-time path query. As most of existing Web mapping services support adding waypoints into a route request, e.g., Google Maps allows up to eight intermediate waypoints in a route request for evaluation users and 23 intermediate waypoints for premier users [9], we attempt to select optimal waypoints in a route request to further reduce the number of external requests and the query response time to users. We first model the problem of selecting

optimal waypoints in a road network, i.e., the route through those waypoints can be shared by the largest number of queries, as the problem of finding the longest simple path in a weighted bipartite digraph. Since it is a NP-complete problem, we propose a greedy algorithm to find the best set of waypoints in an external route request, the time complexity of which is only $O(mn)$, where m is the maximum allowed number of waypoints in a route request and n is the number of vertices in a graph (m is usually much smaller than n).

To evaluate the performance of our proposed algorithm, we compare it with two basic algorithms using real Web mapping services, a real road network, real and synthetic data sets. Experimental results show that our proposed algorithm is efficient and scalable with various numbers of queries and waypoints.

2 Related Work

Existing shortest path query processing algorithms in road networks can be categorized into two main models. (1) ***Time-independent road networks.*** In this model, shortest path query processing algorithms assume that the cost or weight of a road segment, which can be in terms of distance or travel time, is constant (e.g., [10–12]). These algorithms mainly rely on pre-computed distance or travel time information of road segments in road networks. However, the actual travel time of a road segment may vary significantly during different time of a day due to dynamic traffic on road segments [13]. (2) ***Time-dependent road networks.*** The shortest path query processing algorithms designed for this model have the ability to support the dynamic weight of a road segment and topology of a road network, which can change with time. This model is more realistic but more challenging. George et al. [14] proposed a time-aggregated graph, which uses time series to represent time-varying attributes. The time-aggregated graph can be used to compute the shortest path at a given start time or to find the best start time for a path that leads to the shortest travel time. In time-dependent road networks where the weight of each road segment is a function of time, problems of the time-dependent shortest-path [13, 15] also have been extensively studied. The solutions for those problems may capture the effects of periodic events (e.g., rush hours, and weekdays). However, they still cannot reflect live traffic information, which can be affected by sudden events, e.g., congestions, accidents, and road maintenance.

This paper focuses on the shortest travel-time route (or path) queries in time-dependent road networks. Our work distinguishes itself from previous work in that it does not model the underlying road network based on different criteria [13–15]. Instead, it employs external Web mapping services, e.g., Google Maps, to provide the real-time route information in a road network.

There are some work about query processing with expensive attributes that are accessed from external Web services (e.g., [1–4, 8, 16, 17]). To minimize the number of external requests, [8, 16, 17] mainly focus on using either some *cheap* attributes that can be retrieved from local data sources [8, 16] or sampling methods [17] to prune a whole set of objects into a smaller set of candidate objects;

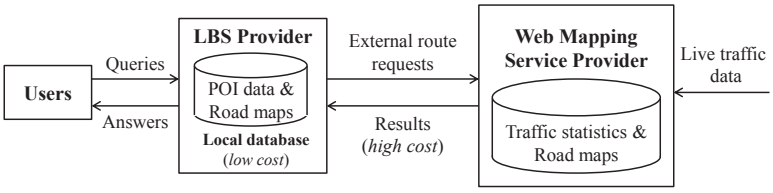


Fig. 1. System architecture.

then, they only issue necessary external requests for retrieving candidate objects. The closest work to our paper are [1–4]. In [1–3], the authors proposed k-NN query processing algorithms that utilize grouping, direction sharing, shared query execution, pruning techniques and parallel requesting to reduce the number of external requests to Web mapping services and provide highly accurate query answers. In [4], route logs are employed to derive tight lower/upper bounding travel times to reduce the number of external Web mapping requests for answering range and k-NN queries. However, none of these existing techniques focuses on how to utilize waypoints in an external route request to reduce the number of such expensive requests, which is the problem that we address in this paper.

3 System Model

In this section, we describe our system architecture, road network model, problem definition, and the objectives of the paper.

System architecture. Figure 1 depicts the system architecture that consists of three entities: users, an LBS provider, and a Web mapping service provider (e.g., Google Maps). Users send shortest travel-time path queries with query origins and destinations to the LBS provider, where the origin could be a user's current location or any other location and the destination could be any POI or location that the user wants to go or is recommended by the LBS provider. Besides turning in its own business information to a querying user, the LBS provider also returns the detailed direction information of the shortest route based on the user selected travel model (e.g., driving, walking, or cycling), which is accessed from the Web mapping service provider.

Road network model. In this paper, a road map is modeled as a graph $G = (V, E)$ comprising a set V of vertices with a set E of edges, where each road segment is an edge and each intersection of road segments is a vertex. As the weight of each edge is static (i.e., the network distance), G cannot provide real-time travel-time-based route information.

Problem definition. Given a set of user queries $\mathcal{Q} = \{q_1, q_2, \ldots, q_n\}$ arrived concurrently or within a short time period at the LBS provider, and each user query q_i is in a form of (o_i, d_i), where o_i and d_i are query origin and destination, respectively, the LBS provider returns the shortest travel-time route with the

detailed direction information from o_i to d_i for each user query q_i based on live traffic conditions, which are provided by the Web mapping service provider.

Our objectives. Since there are two critical limitations to access the travel route information from a Web mapping service as stated above, i.e., high cost and usage limits, our objectives are to reduce the number of external Web mapping requests issued by the LBS provider and the query response time to its querying users.

4 Direction Sharing Optimization

For an external route request with an origin and a destination, the Web mapping service will return the shortest travel-time route from the origin to the destination with detailed turn-by-turn direction information. Shortest routes exhibit the optimal sub-route property [2, 18, 19], i.e., every sub-route of the shortest route is also the shortest sub-route. In other words, the shortest route $R(o_i \rightarrow d_i)$ returned from the Web mapping service for a query $q_i = (o_i, d_i)$ can be shared used by another query $q_j = (o_j, d_j)$, if both o_j and d_j are located in the route in order[1], i.e., $o_j \in R(o_i \rightarrow d_i)$ and $d_j \in R(o_i \rightarrow d_i)$; thus, there is no need for the LBS provider to issue an external Web mapping request for $q_j = (o_j, d_j)$ any more. As a result, the number of external requests can be reduced.

To reflect the possibility of sharing the route information of a route with others, we formally define its *sharing ability* as follows:

Definition 1 (Sharing Ability). *Let $\mathcal{Q}$ be a set of user queries, and $R(o_i \rightarrow d_i)$ be the shortest travel-time route for query $q_i = (o_i, d_i)$ ($q_i \in Q$). The* sharing ability *of $R(o_i \rightarrow d_i)$ with respect to $\mathcal{Q}$ (denoted as $SA(R(o_i \rightarrow d_i), \mathcal{Q})$) is the number of queries in $\mathcal{Q}$, where the direction information of $R(o_i \rightarrow d_i)$ can be shared with these queries.*

Since the *sharing ability* can reflect the possibility of sharing the direction information of a route with others, to minimize the number of external route requests to the Web mapping service, the LBS provider should process queries in $\mathcal{Q}$ in a non-increasing order based on their *sharing abilities* to fully utilize the direction sharing optimization. However, the direction information between two locations for a query is unknown until the LBS provider issues an external route request to the Web mapping service and gets its result. Only knowing the road network model G and the origins and destinations of queries in $\mathcal{Q}$, it would be impossible to calculate the exact *sharing ability* of a route. To this end, we utilize the network distance of a route to approximately reflect its *sharing ability*, i.e., a route with a longer network distance has a higher *sharing ability*.

5 Route Waypoint Optimization

Most of existing Web mapping services allow users to specify waypoints in a route request. Waypoints alter a route by routing it through the specified locations. For example, given a route request with a set of ordered waypoints

[1] In this paper, we assume that a route is directional.

$\mathcal{W} = \{w_1, w_2, \cdots, w_m\}$, where w_1 and w_m are the origin and destination locations, respectively, and the others are intermediate locations, Web mapping services will return the detailed direction information from w_1 to w_2 to $\cdots$ to w_m as a result for the route request (i.e., $R(w_1 \rightarrow w_2 \rightarrow \cdots \rightarrow w_m)$), and the sub-route $R(w_i \rightarrow w_{i+1})$ (where $1 \leq i \leq m-1$) between any two successive waypoints in the result is with the shortest travel time.

However, most of queries issued by users only seek for the shortest travel-time route information from their own origins to corresponding destinations, i.e., only two waypoints, which is less than the maximum allowed number of waypoints provided by Web mapping service providers. For example, Google Maps allows up to 8 intermediate waypoints in a route request for evaluation users and 23 intermediate waypoints for premier users [9]. On the other hand, there is usually a limit or charge on the number of requests to a Web mapping service.

It is unwise for the LBS provider to simply issue external route requests to the Web mapping service provider for each query in $\mathcal{Q}$ separately without utilizing waypoints. Thus, a more efficient way is composing multiple user queries in $\mathcal{Q}$ as one external route request by taking the origins and destinations of those queries as waypoints.

5.1 Problem Modeling

To minimize the number of external route requests, the LBS provider should make full utilization of the direction sharing and waypoints provided by Web mapping services. The optimal ordered waypoint set $\mathcal{W} = \{w_1, \cdots, w_m\}$ should be determined for an external route request to maximize the possibility of sharing its direction information with other queries in $\mathcal{Q}$. In other words, we should maximize the *sharing ability* of $R(w_1 \rightarrow \cdots \rightarrow w_m)$ with respect to the queries in $\mathcal{Q}$ by maximizing the *sharing ability* of the sub-routes in $\mathcal{W}$ with respect to the queries in $\mathcal{Q}$; hence,

$$SA(R(w_1 \rightarrow \cdots \rightarrow w_m), \mathcal{Q}) = \sum_{i=1}^{m-1} SA(R(w_i \rightarrow w_{i+1}), \mathcal{Q}). \quad (1)$$

To find the optimal $R(w_1 \rightarrow \cdots \rightarrow w_m)$, we first model the origins and destinations of the queries in $\mathcal{Q}$ and possible routes between them as a weighted semicomplete bipartite digraph $G_w = (O_w, D_w, E_w)$ comprising a set O_w of origin vertices, a set D_w of destination vertices, and a set E_w of weighted directed edges. The vertex set is composed by two disjoint sets O_w and D_w, where O_w and D_w consist of the origins, i.e., o_i $(i = 1, 2, \cdots, n)$, and destinations, i.e., d_i $(i = 1, 2, \cdots, n)$, of the queries in $\mathcal{Q}$, respectively. The set E_w consists of two parts of edges: (1) the edges starting from the query origins and ending at their corresponding destinations, i.e., $o_i d_i$ $(i = 1, 2, \cdots, n)$, and (2) the edges starting from the query destinations and ending at all other origins except for their corresponding origins, i.e., $d_i o_j$ $(i, j = 1, 2, \cdots, n$ and $i \neq j)$. The weight of each edge in E_w is represented by the *sharing ability* of its corresponding route, e.g., $weight(d_i o_j) = SA(R(d_i \rightarrow o_j), \mathcal{Q})$.

5.2 Waypoint Selection

After modeling the queries in $\mathcal{Q}$ as a weighted bipartite digraph $G_w = (O_w, D_w, E_w)$, we should find a waypoint set $\mathcal{W} = \{w_1, \cdots, w_m\}$ for $\mathcal{Q}$ such that the route $R(w_1 \rightarrow \cdots \rightarrow w_m)$ will result in the best *sharing ability*. We can transform this problem into how to find a simple path in $G_w = (O_w, D_w, E_w)$ with the largest weight via at most m vertices.

Since finding the longest simple path in a weighted graph is a NP-complete problem [18, 20], in this work, we aim to find the best set of waypoints in an external route request for all the queries in $\mathcal{Q}$ by designing a greedy algorithm that attempts to find the best path in $G_w = (O_w, D_w, E_w)$. The time complexity of the proposed greedy algorithm is only $O(mn)$, where m is the maximum allowed number of waypoints in a route request supported by Web mapping services, n is the number of vertices in G_w, and m is usually much smaller than n.

The main idea of the greedy algorithm is to select the edge with the largest weight (i.e., the best *sharing ability*) from E_w, and then insert the vertices of the selected edge to waypoint set $\mathcal{W}$. The vertices that are connected to the selected edges are considered as waypoints in a route request. The details of the greedy algorithm are as follows.

The greedy algorithm. Given a query set $\mathcal{Q}$ and the maximum number of waypoints m, the proposed greedy algorithm consists of two main steps:

1. **Step 1: Initial waypoint selection step.** For each query $q_i = (o_i, d_i)$ in $\mathcal{Q}$, the *sharing ability* of its query route with respect to $\mathcal{Q}$ (i.e., $SA(R(o_i \rightarrow d_i), \mathcal{Q})$ is calculated, and then the origin o_l and destination d_l of the query q_l with the highest *sharing ability* are selected. o_l is inserted into $\mathcal{W}$ as the first waypoint, and it is removed from the query origin set O_w. If the number of waypoint in $\mathcal{W}$ (i.e., $|\mathcal{W}|$) is less than m, d_l is also inserted into $\mathcal{W}$ as the second waypoint and removed from the query destination set D_w. The algorithm proceeds to the next step if $|\mathcal{W}| < m$ and O_w is not empty.
2. **Step 2: Waypoint selection step.** In this step, d_l is the *last* waypoint inserted into $\mathcal{W}$. The next waypoint o_t is selected from all the origins in O_w and inserted into $\mathcal{W}$ such that $R(d_l \rightarrow o_t)$ has the highest *sharing ability*. o_t is then removed from O_w. If $|\mathcal{W}| < m$, the corresponding destination d_t of o_t is also added to $\mathcal{W}$, and d_t is removed from D_w. This step is repeated until $|\mathcal{W}| = m$ or both the origin and destination sets (i.e., O_w and D_w) become empty.

Finally, a route request with the waypoints in $\mathcal{W}$ is sent to the Web mapping service to retrieve the required direction information for the queries in $\mathcal{Q}$.

5.3 Algorithm

Given a query set $\mathcal{Q} = \{q_1, q_2, \ldots, q_n\}$ and the maximum number of waypoints m, Algorithm 1 shows the pseudo code of the algorithm with the direction sharing and route waypoint optimizations. After we find the waypoint

Algorithm 1 The algorithm with the direction sharing and route waypoint optimizations.

```
1: input: query set Q and the maximum number of waypoints m
2: while Q is not empty do
3:    Find the waypoint set W = {w_1, ..., w_m} by the greedy algorithm;
4:    Retrieve R(w_1 → ... → w_m) by issuing a route request to the Web mapping
      service;
5:    for each query q_i = (o_i, d_i) in Q do
6:       if o_i and d_i is located in any R(w_i → w_{i+1}) (i = 1, ..., m − 1) then
7:          Use the retrieved direction information to answer q_i;
8:          Remove q_i from Q;
9:       end if
10:   end for
11: end while
```

set $\mathcal{W} = \{w_1, \cdots, w_m\}$ by the greedy algorithm, the LBS provider issues an external route request with the ordered waypoints in $\mathcal{W}$ to retrieve the direction information of $R(w_1 \rightarrow \cdots \rightarrow w_m)$ (Lines 3 to 4). For each query $q_i = (o_i, d_i)$ in $\mathcal{Q}$, if o_i and d_i is located in any $R(w_i \rightarrow w_{i+1})$ $(i = 1, \cdots, m-1)$, the algorithm can use the retrieved direction information to answer q_i and q_i is removed from $\mathcal{Q}$ (Lines 7 to 8). The algorithm repeatedly deals with the remaining queries in $\mathcal{Q}$ until $\mathcal{Q}$ becomes empty.

6 Performance Evaluation

In this section, we first give our evaluation model in Section 6.1, including basic approaches, performance metrics, and experiment settings. Then, we analyze the experimental results in Section 6.2.

6.1 Evaluation model

Basic approaches. To the best of our knowledge, there is no other existing work about taking advantage of waypoints in route requests to reduce the number of external Web mapping requests for spatial query processing. To this end, we design two baseline algorithms to show the effectiveness of our proposed approach.

The direction sharing optimization has been proposed in our previous work [2] that is considered as a baseline algorithm (denoted as DS). The basic idea of DS is that: (1) the LBS provider processes the queries in $\mathcal{Q}$ based on the non-ascending order of their network distance in G by issuing external requests to the Web mapping service provider. (2) When a travel route is returned, its direction information is used to answer the queries with origins and destinations located in the route and these queries are removed from $\mathcal{Q}$. DS repeatedly processes the remaining queries in $\mathcal{Q}$ based on these two steps until $\mathcal{Q}$ becomes empty.

To reflect the effectiveness of our proposed greedy algorithm that aims to select the best set of waypoints in an external route request, (denoted as Greedy),

we design another baseline algorithm by randomly selecting waypoints from the sets of query origins and destinations (denoted as Random). The basic idea of Random is that: (1) the LBS provider randomly selects waypoints from O_w and D_w in G_w into $\mathcal{W}$. (2) The LBS provider next issues an external route request with the waypoints in $\mathcal{W}$ to retrieve the direction information of $R(w_1 \rightarrow \cdots \rightarrow w_m)$. (3) When the direction information of the route is returned, it is used to answer the the queries with their origins and destinations located in any $R(w_i \rightarrow w_{i+1})$ $(i = 1, \cdots, m-1)$ and these queries are removed from $\mathcal{Q}$. These three steps are repeatedly executed until $\mathcal{Q}$ becomes empty.

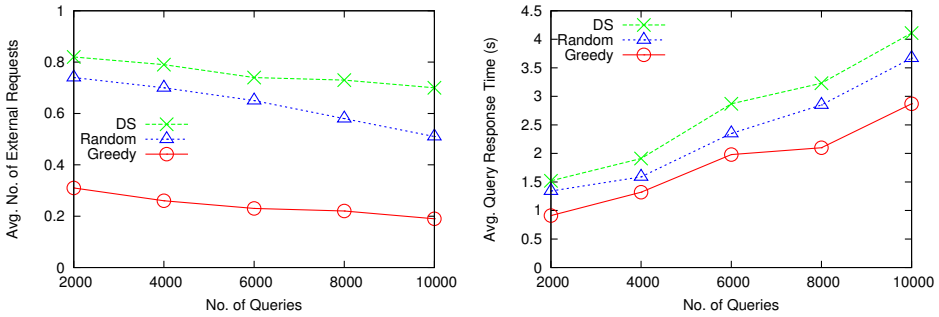
Performance metrics. We evaluate the performance of the basic approaches (i.e., DS and Random) and our proposed algorithm (i.e., Greedy) in terms of two metrics: (1) the average number of external route requests submitted to the Web mapping service per user query and (2) the average query response time per user query. The query response time of a query is the time from the time when the query is received by the LBS provider to the time when the answer is returned to the querying user.

Experiment settings. We evaluated the performance of DS, Random, and Greedy using C++ with a real road network of Hennepin County, Minnesota, USA. We selected an area of 8×8 km^2 that contains 6,109 road segments and 3,593 intersections, and the latitude and longitude of its left-bottom and right-top corners are (44.898441, -93.302791) and (44.970094, -93.204015), respectively. For all queries in our experiment, the query origins are randomly generated and the query destinations are randomly selected from a real POI dataset (including 126 bars, 320 restaurants, and 491 food places), which is collected from Google Places API [21] within the selected area. MapQuest Maps is taken as the Web mapping service in our experiments.

6.2 Experimental results

We evaluate the scalability, efficiency, and applicability of the proposed algorithm (i.e., Greedy) and the basic approaches (i.e., DS and Random) with respect to various numbers of queries and waypoints.

Effect of the number of queries In this section, the number of waypoints provided by the Web mapping service is set to six (i.e., $m = 6$), and the number of queries varies from 2,000 to 10,000 to evaluate the performance of DS, Random, and Greedy with respect to various numbers of queries as depicted in Figure 2. Without any optimization, the LBS provider needs to issue one external request for each query. With the direction sharing optimization, the average number of external requests of DS decreases to less than one request, i.e., nearly 20% reduction (Figure 2a). With the utilization of waypoints, the average number of external route requests further reduces, i.e., nearly 40% and 75% are reduced by Random and Greedy, respectively. These experimental results show the effectiveness of Greedy for the waypoint section as it performs much better than Random. When the number of queries gets larger, there are more outstanding



(a) Average number of external route requests

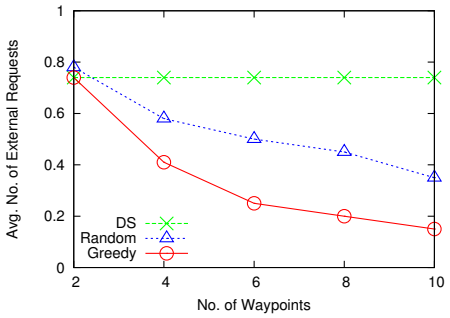
(b) Average query response time

Fig. 2. Effect of the number of queries.

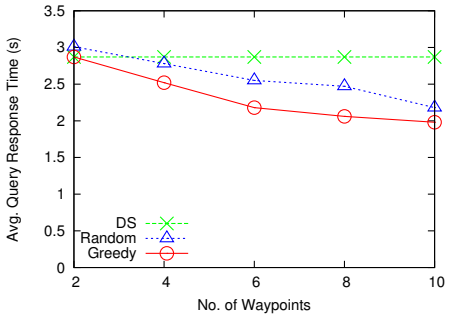
queries in the road network; hence, the direction information of a route has a higher chance to be shared with more queries (i.e., the power of the direction sharing optimization increases); thus, the number of external route request gets lower for all evaluated algorithms, as show in Figure 2a. The results also show the scalability of Greedy with the increase of the number of queries.

In all the experiments, we do not consider parallel requesting to the Web mapping service provider, i.e., the LBS provider issues external route requests to the Web mapping service in a sequential manner. When there are more queries arrived, a query encounters a longer waiting time. Therefore, the average query response time gets longer with the increase of the number of queries, as depicted in Figure 2b. Greedy yields the least average query response time compared with the other two basic algorithms, which benefits from the direction sharing and route waypoint optimizations.

Effect of the number of waypoints in an external route request In this section, the number of queries is set to 6,000, and the number of waypoints provided by the Web mapping service provider varies from 2 to 10, to study the performance of DS, Random, and Greedy with respect to various numbers of waypoints, as depicted in Figure 3. When the number of waypoints gets larger, the average number of external route requests for DS remains the same, while that for Random and Greedy drops significantly (Figure 3a). This is because both Random and Greedy employ the route waypoint optimization while DS does not. With more waypoints utilized in an external route request, its direction information can be used to answer more queries. Figure 3a also shows that Greedy is the most effective algorithm with different numbers of waypoints as it consistently yields less external requests than Random. Similarly, the average query response time of DS is not affected by the number of waypoints in an external route request, while that of Random and Greedy drops gradually along with the increase of the number of waypoints (Figure 3b), as Random and Greedy



(a) Average number of external route requests



(b) Average query response time

Fig. 3. Effect of the number of waypoints in an external route request.

can utilize more waypoints in one route request. As a result, our Greedy can achieve not only the best number of external route requests but also the shortest query response time compared with the other two basic algorithms.

7 Conclusion

In this paper, we have proposed a spatial mashup framework for a location-based service (LBS) provider to retrieve travel route information between two location points in a road network through issuing an external route request to a Web mapping service, e.g., Google Maps, Bing Maps, and MapQuest Maps. Since retrieving external data is much more expensive than accessing local data, we employ the direction sharing and route waypoint optimizations to reduce the number of such external route requests. To reflect the possibility of sharing the direction information of a route, we first formally define a method to measure its *sharing ability*. Then, we model the problem of selecting the optimal set of waypoints in a road network for an external route request, i.e., the *sharing ability* of the sub-routes in a route request should be maximized, as the problem of finding the longest simple path in a weighted bipartite digraph. We design a greedy algorithm that aims to find the best waypoint set for an external route request. We evaluate the performance of our algorithm using real Web mapping services, a real road network, real and synthetic data sets. Our experimental results show that our proposed algorithm is efficient and scalable with various numbers of queries and waypoints in a route request.

Acknowledgments

This work was supported in part by the Fundamental Research Funds for the Central Universities in China (Project No. JUSRP11557), the National Natural Science Foundation of China (Project No. 61572336).

References

1. Zhang, D., Chow, C.Y., Li, Q., Zhang, X., Xu, Y.: Efficient evaluation of k-NN queries using spatial mashups. In: SSTD. (2011)
2. Zhang, D., Chow, C.Y., Li, Q., Zhang, X., Xu, Y.: SMashQ: Spatial mashup framework for k-NN queries in time-dependent road networks. Distributed and Parallel Databases, DAPD **31**(2) (2013) 259–287
3. Zhang, D., Chow, C.Y., Li, Q., Zhang, X., Xu, Y.: A spatial mashup service for efficient evaluation of concurrent k-nn queries. IEEE Transactions on Computers accepted to appear
4. Li, Y., Yiu, M.L.: Route-saver: Leveraging route apis for accurate and efficient query processing at location-based services. IEEE TKDE **27**(1) (2015) 235–249
5. ProgrammableWeb: http://www.programmableweb.com/category-api
6. The Google Directions API: https://developers.google.com/maps/documentation/directions
7. MapQuest Directions Web Service: http://www.mapquestapi.com/directions
8. Levandoski, J.J., Mokbel, M.F., Khalefa, M.E.: Preference query evaluation over expensive attributes. In: CIKM. (2010)
9. Google Maps/Google Earth APIs Terms of Service: http://code.google.com/apis/maps/terms.html
10. Wu, L., Xiao, X., Deng, D., Cong, G., Zhu, A.D., Zhou, S.: Shortest path and distance queries on road networks: An experimental evaluation. In: VLDB. (2012)
11. Zhu, A.D., Ma, H., Xiao, X., Luo, S., Tang, Y., Zhou, S.: Shortest path and distance queries on road networks: towards bridging theory and practice. In: ACM SIGMOD. (2013)
12. Sommer, C.: Shortest-path queries in static networks. ACM Computing Surveys, CSUR **46**(4) (2014) 45
13. Demiryurek, U., Banaei-Kashani, F., Shahabi, C., Ranganathan, A.: Online computation of fastest path in time-dependent spatial networks. In: SSTD. (2011)
14. George, B., Kim, S., Shekhar, S.: Spatio-temporal network databases and routing algorithms: A summary of results. In: SSTD. (2007)
15. Ding, B., Yu, J.X., Qin, L.: Finding time-dependent shortest paths over large graphs. In: EDBT. (2008)
16. Bruno, N., Gravano, L., Marian, A.: Evaluating top-k queries over web-accessible databases. In: IEEE ICDE. (2002)
17. Chang, K.C.C., Hwang, S.W.: Minimal probing: Supporting expensive predicates for top-k queries. In: ACM SIGMOD. (2002)
18. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms (3. ed.). MIT Press (2009)
19. Thomsen, J.R., Yiu, M.L., Jensen, C.S.: Effective caching of shortest paths for location-based services. In: ACM SIGMOD. (2012)
20. Karger, D., Motwani, R., Ramkumar, G.: On approximating the longest path in a graph. Algorithmica **18**(1) (1997) 82–98
21. The Google Places API: https://developers.google.com/places/